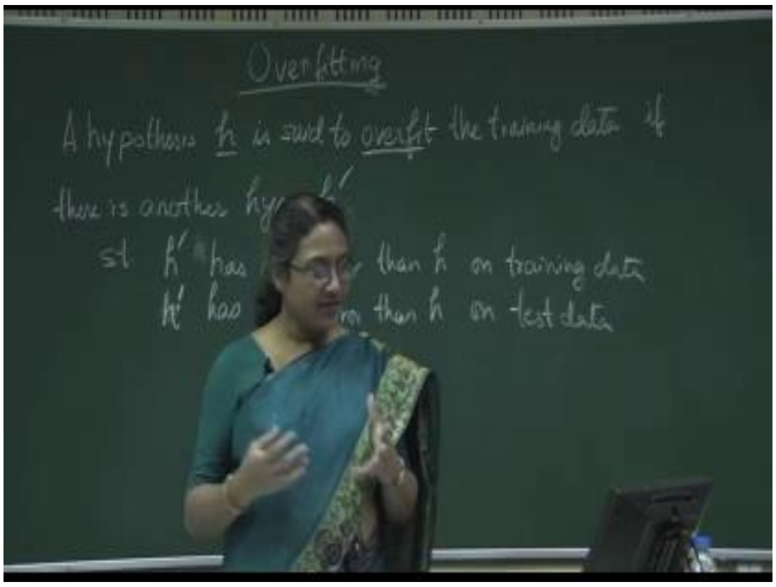
Introduction to Machine Learning
Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 08
Overfitting

Good morning, welcome to the forth part of Module 2. Today, we will discuss about Overfitting. So, we mentioned overfitting when we talked about a decision tree in passing. So, first we will define what is overfitting.

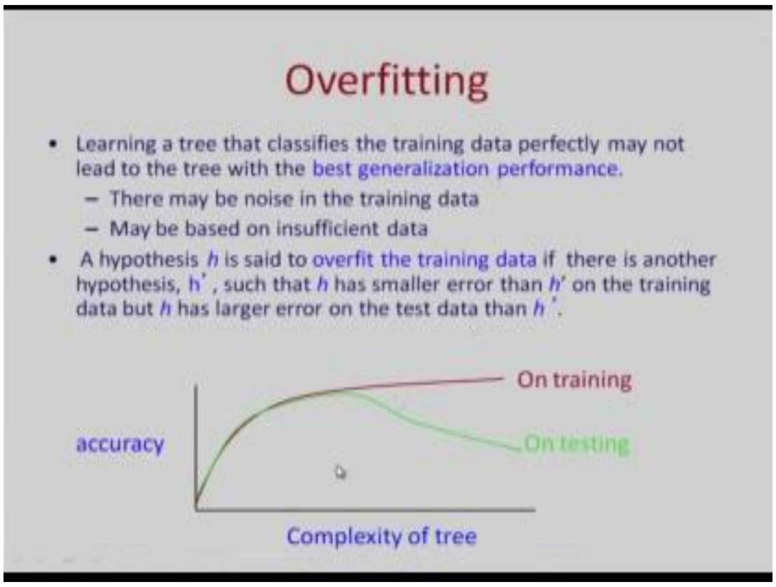
If you learn decision tree and you continue splitting nodes, the tree will becomes larger and larger. When the tree becomes larger, if you use the tree for classifying the training examples the error on the training examples will reduce, but it may happen that after sometime as you grow the tree, the true error will increase. So, this is called overfitting. Let us formally define overfitting.

(Refer Slide Time: 01:11)


Overfitting
A hypothesis h is said to overfit the training data if
there is another hyp h′
st h′ has than h on training data
h′ has than h on test data

We say that a hypothesis ‘h’ is said to overfit the training data if there is another hypothesis h prime, such that h prime has more error than h on training data, but h prime has less error than h on test data. For example, in the context of decision trees if it happens that we have a smaller decision tree and it has a higher error on training data, and lower error on test data compared to a larger decision tree which has smaller error on training data, but higher error on test data, we say overfitting has occurred.

(Refer Slide Time: 03:18)



Overfitting

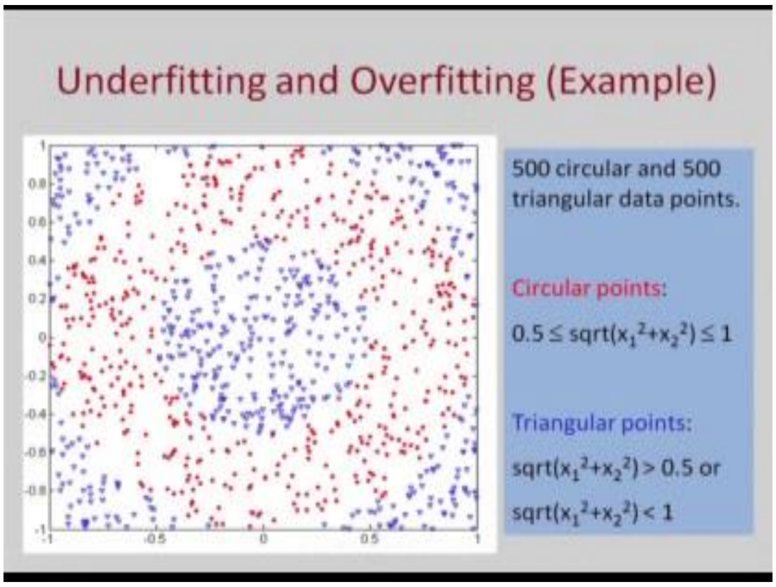
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h', such that *h* has smaller error than *h'* on the training data but *h* has larger error on the test data than *h'*.

So, let us look at this slide, that suppose as we are growing a decision tree and as we are splitting the nodes, we are splitting the nodes when that is positive information gain that is there is a result and production in entropy. Typically what we expect as we keep growing the tree, the accuracy of the tree will grow on increasing on the training set that is the error will go on decreasing on the training set.

However, if you test the tree at different points in the algorithm on a held out test set, disjoint test set what we typically observe is that as you grow the tree, initially accuracy on the test set increases after sometime the accuracy on the test set starts to fall. Now, why and this happen? You see that we want to have good generalization of forms, but the training data may contain noise and as we make the decision tree more and more complex it may try to fit noise and therefore, the resulting decision tree will not work well on the true population.

A second reason could be you have two few training examples and you are trying to accommodate the errors in the training examples and you do badly on the test set, this is why overfitting can occur in decision tress.

(Refer Slide Time: 05:00)



Underfitting and Overfitting (Example)

500 circular and 500 triangular data points.

Circular points:

$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

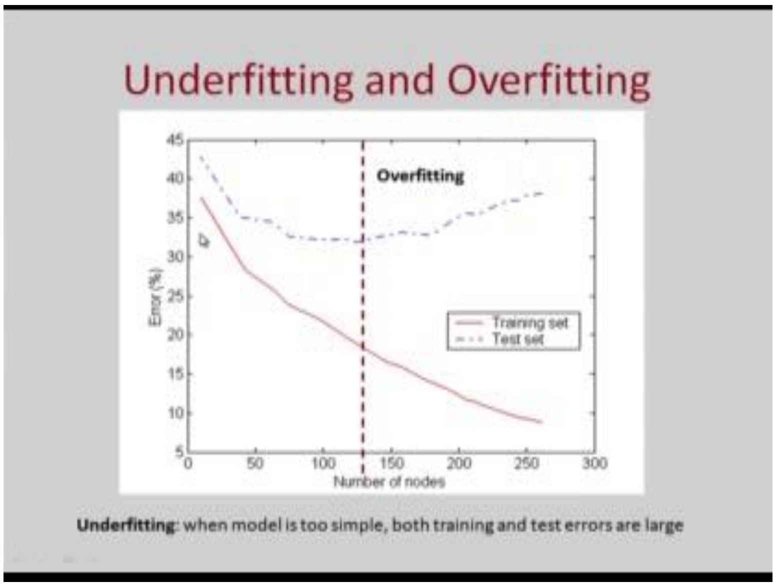
Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

$\text{sqrt}(x_1^2+x_2^2) < 1$

Now, to illustrate overfitting there was an experiment which was run where some positive and negative examples where taken, and they were generated such that all points with radius from 0 to 0.5 where positive where blue plus and those having radius from the origin between 0.5 and 1 then red circles and beyond 1 also it was blue triangle. So, this sort of examples where used to construct a decision tree and in this decision tree what we observed is that the error on the training set, the last curve we were plotting accuracy.

(Refer Slide Time: 05:51)



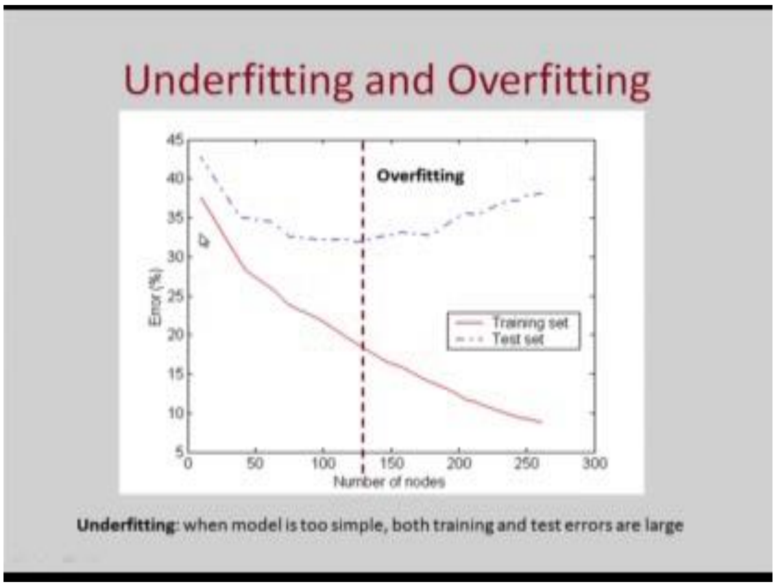
Underfitting and Overfitting

Underfitting: when model is too simple, both training and test errors are large

So, accuracy was going on increasing as we increase the size of the tree. Here, we are plotting error. So, error on the training set will go on reducing as we construct the decision tree, but if we check the error on the test set, initially the error on the test set decreases after that it increases and where it starts to increase in this region we say overfitting has occurred.

In this region, where the tree is not big enough we say there is underfitting. In this region on the left that is underfitting, in the middle there is a good fit and on the right of this dashed line we say overfitting as occurred. So, underfitting occurs when the model is too simple and overfitting occurs when the model is too complex and here is another slide to illustrate overfitting.

(Refer Slide Time: 06:51)

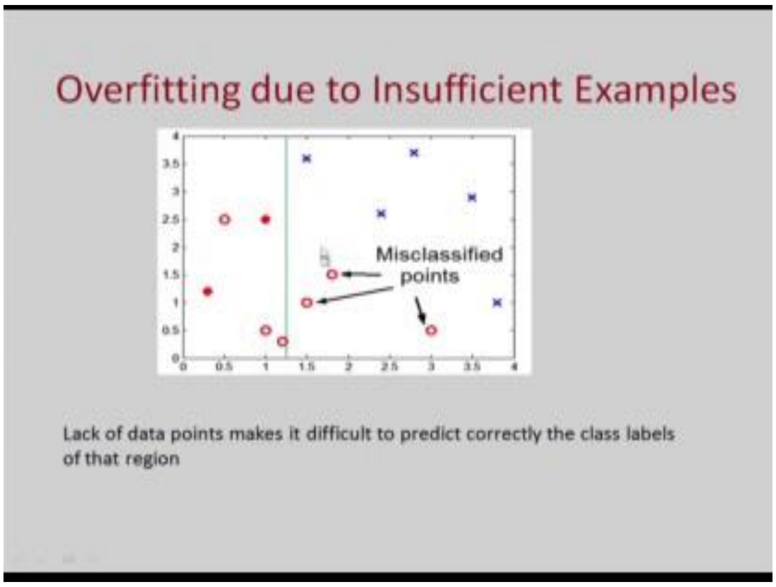


Underfitting and Overfitting

Underfitting: when model is too simple, both training and test errors are large

Now, when you have overfitting what will happen is that, suppose we can take another examples. Suppose, these are plus is positive point and these red circles are negative point and suppose there is a noise point here. If we have this noise point, which is a spurious point, and if you really want to have the decision tree to have no error than the decision tree will try to include the noise point and if it includes the noise point, it will induce error in true set and overfitting can also be due to not enough examples.

(Refer Slide Time: 07:35)



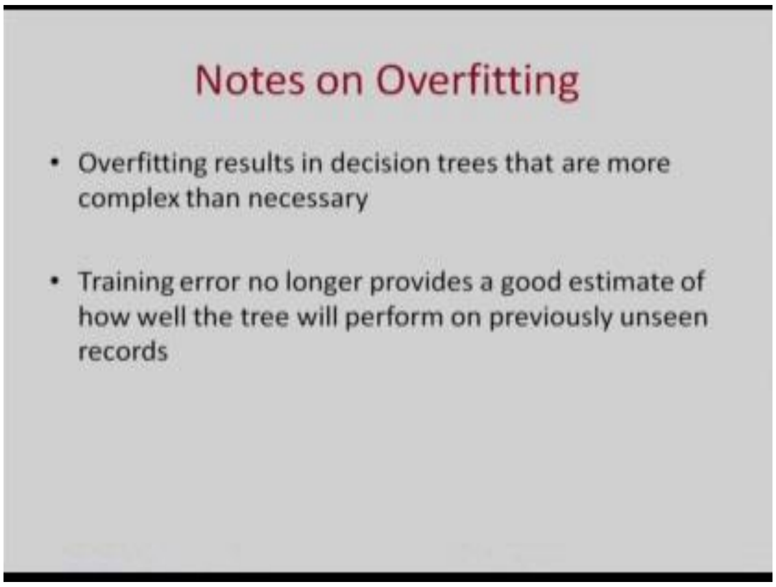
Overfitting due to Insufficient Examples

Lack of data points makes it difficult to predict correctly the class labels of that region

For example, if you look at the slide the bottom portion we do not have too few examples and because we do not have too few examples, this green line is the ideal. So, this are crosses, blue crosses are the one class and the red solid circles is the other class.

And the red hollow circles are your test points because you do not have too few, too enough examples these points may be misclassified. If in a region, in the feature space you do not have sufficient examples then you cannot form a proper hypothesis about that region and this may lead to overfitting. So, if you are working with decision trees, overfitting results in a more complex decision tree that is necessary and training error is not a good predictor for test error. So, what you have to do; training error is not a good predictor for true error.

(Refer Slide Time: 08:42)

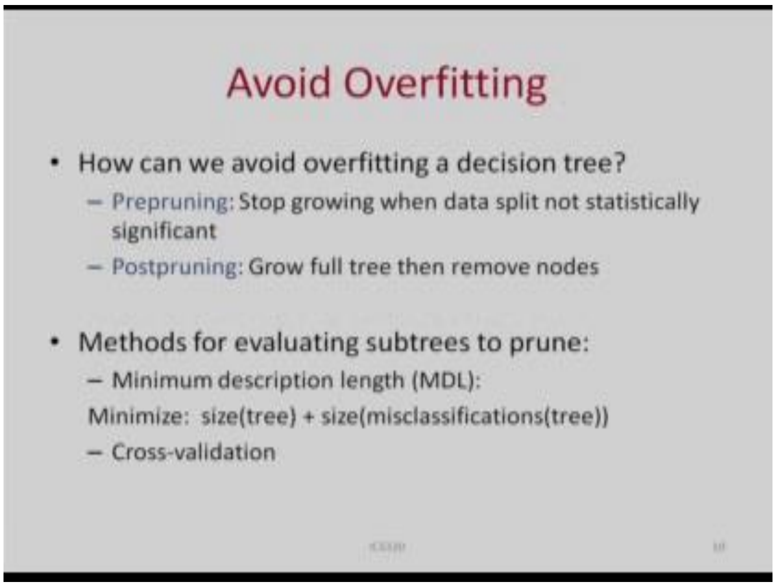


Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

And therefore, you should use a separator unseen sample call the test sample in order to get a better estimate of the true error. We have to now see now that we know that overfitting has occurred we have to look at what can be done to avoid overfitting.

(Refer Slide Time: 09:05)



Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes
- Methods for evaluating subtrees to prune:
  - Minimum description length (MDL):

    Minimize: size(tree) + size(misclassifications(tree))
  - Cross-validation

Now, let us look in the context of decision trees. There are two types of methods that are used to prevent overfitting. So, first of all you could prune the decision tree while growing, you can stop early. So, you have pre-pruning type of methods and you have post-pruning type of methods.

(Refer Slide Time: 09:30)

Overfitting
Pre-pruning : Stop early – Stop splitting when gain is not statistically significant
Post-pruning. Grow and remove nodes
T
MDL
cross validation

So, pre-pruning is a methods pruning decision trees to deal with overfitting and what pre-pruning involves is stopping early and when do you stop? You stop when that data split is not statistically significant. What you saw with the entropy measure is that we split a node on an attribute and we compute the resulting entropy. So, there is S is the examples here, S 1 and s 2 is the set of examples here and we find the gain of S with A, and we find the attribute with the highest gain. The attribute with a highest gain should have a positive gain if there is a negative gain there is no reason to use the attribute, but if the even if there is a positive gain if the gain is not significant, if the gain is not statistically significant we will stop at the node. So, you stop early that is stop growing, stop splitting when gain is not statistically significant that is you do not grow the full tree.

A second type of method of pruning decision trees is based on post-pruning. In post-pruning what you do is you continue growing the tree till the tree is quite large, but after growing a big tree you pruned different sub trees of the tree. So, grow the full tree and remove the nodes. Now, how to remove the nodes? There are several heuristic which have been used to decide which sub trees to remove. So, you may decide to say that, suppose you have grown a decision trees which is quite big.

Suppose, this is some decision tree that you have grown and you want to decide whether you want to prune the sub tree that is you want to make this as a leaf node now what you can do is that you can use cross validation. So, what you can do is you can use a validation set and you check the error of the original tree and suppose this is the sub tree st 1. So, you find out the error you find out the error of T and error of T minus st 1, right

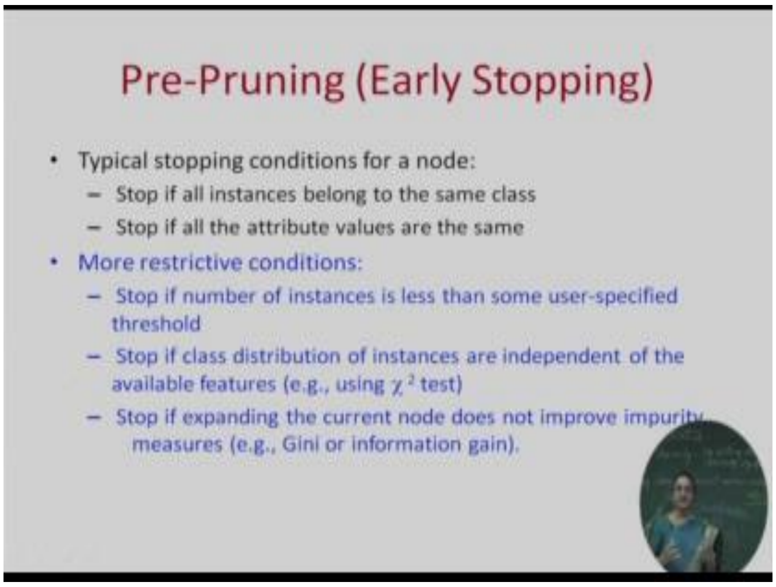
So, you find out the error after removing the sub tree, if this error is smaller than this is a candidate for removal. Now, when you have a decision tree any internal node is a candidate. For removal you can remove this node, this node, this node, this node, this node, this node these are different or even this node. So, these are the different candidates sub trees for removal and among them you want to find out those sub trees whose removal improves the test error. Among those, if there are no tree whose removal improves the test error you do not move further among those tree which are eligible for among those sub trees which are eligible for removal you choose that sub tree whose removal lowers the error the most. So, that is by using cross validation.

Apart from cross validation, there is another method based on a principle called MDL or Minimum Description Length. We will not talk about this principle in detail. The basic idea is you want to look at, you think of having a function or having a decision tree or having a classified as reducing the size of information you know. Suppose, you have a training set and you have a function which classifies the training set perfectly using the training set you can recreate using this function you can recreate the training set given x you can find y, this is the role of this function.

Now, if this function was perfect you could just skip that function you could remove the labels on the training set and you could recreate that, but if your function is not a perfect classifier of these examples what you should do is keep this function and keep the labels of the misclassified examples. You could think of that given a function that total information that you have to keep is the function plus the misclassified examples.

Now, you could compare two different trees, one tree which is smaller and classifies less examples and the second tree which is larger and classifies more examples. You could compare them based on the description length, what is the total in length total number of bits requires to keep the function and the misclassified examples and you want to choose the tree for which this is smallest. That is another principle, but we will not talk about it.

(Refer Slide Time: 16:05)



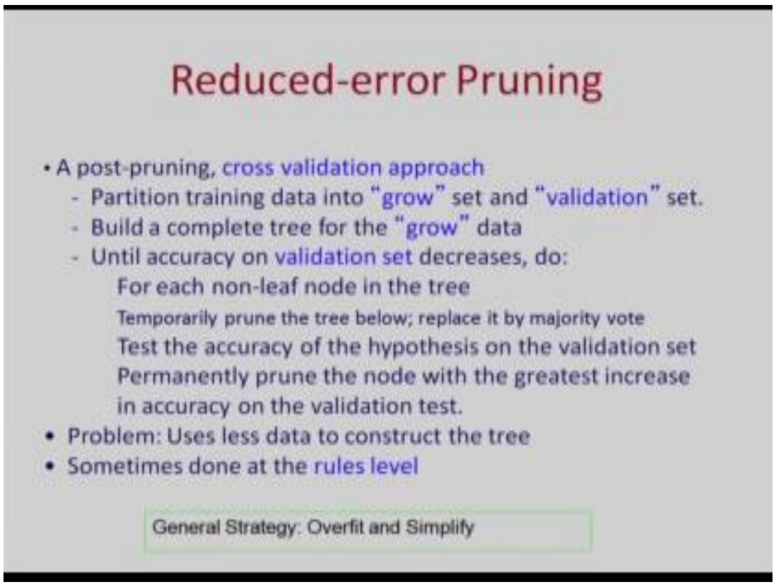
Pre-Pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Let us look at this slide on pre-pruning as we said pre-pruning means early stopping you evaluate the splits before installing them. So, do not install splits that do not look worthwhile, when no worthwhile splits are there to install done. So, typical stopping conditions to node are in a normal decision tree without pruning, you stop if all instances belong to same class. You stop if all instances belong to the have the same value and if you want to use pre-pruning you introduce some more restrictive conditions.

For examples, if you stop if number of instances of that node is too small, you stop if the class distribution are independent of the available feature using some statistic. We will not talk about it in detail. Thirdly you stop expanding the node if the improvement in the information measure is not statistically significant.

(Refer Slide Time: 17:10)



Reduced-error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "grow" set and "validation" set.
  - Build a complete tree for the "grow" data
  - Until accuracy on validation set decreases, do:

```
For each non-leaf node in the tree
Temporarily prune the tree below; replace it by majority vote
Test the accuracy of the hypothesis on the validation set
Permanently prune the node with the greatest increase
in accuracy on the validation test.
```

- Problem: Uses less data to construct the tree
- Sometimes done at the rules level

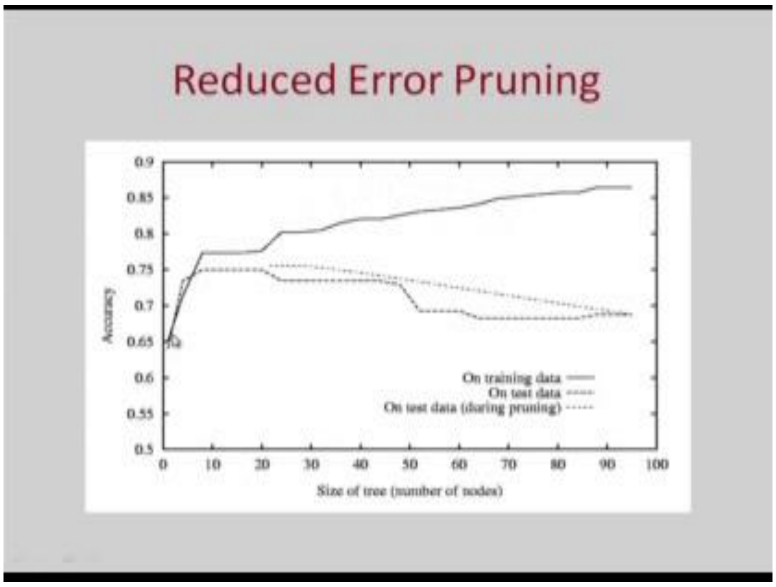
General Strategy: Overfit and Simplify

For post-pruning one of the algorithms, purposed pruning is called reduced-error pruning and we will describe that algorithm. In reduced-error pruning if you look at the slide is a post-pruning cross validation approach. So, given the training set available to us, we will split the training set into a training and validation set or we call this grow set. This part of the training set is used to grow the tree and this part is used to validate the tree. Now, using the grow data we will built a complete which ideally as zero error on the training data if possible otherwise also grow the tree as much as possible.

After that you start pruning the tree and as we said you look at different candidate trees sub trees where pruning and how long will you continue pruning, until you reach a situation where removing any sub tree will give you increased error on the validation set. So, until accuracy on validation set decrease you will do this for each non leaf node in the tree. As we explained any non-leaf node or interior node is candidate for pruning. For any non-leaf node in the tree you temporarily prune the tree below that node, and replace that node by the majority vote test the accuracy of the hypothesis on the validation set and. So, you find out if the resulting error is smaller than the original tree and now you do it for every attribute you prune that sub tree for which this error is lowest.

So, you permanently prune the node with the greatest increase in accuracy on the validation test and you continue doing this until accuracy on the validation set does not increase. This is post-pruning method, there are a few points of this method because you are dividing the data into growth set and validate set you have less number of data on which to grow that.

(Refer Slide Time: 20:03)

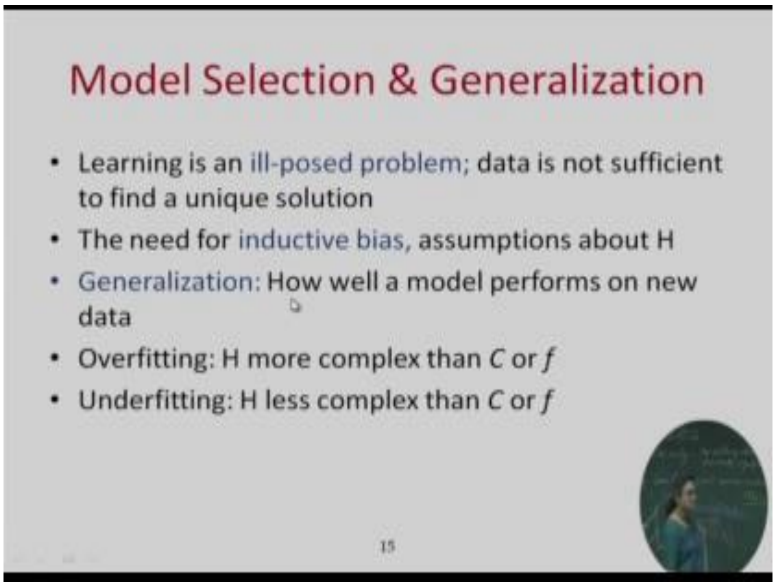


Reduced Error Pruning

Now, if you look at this slide it shows and the effect of post pruning on a particular training set this solid line on the top. This is the training error on the training data as you grow the tree. So, tree is grown to 0, 1 to 10; 100. Up to 100 node tree is grown and as you see the training accuracy is going on increasing, whereas the test data on the validation set the accuracy is reducing after the initially it is increasing after sometime it is reducing that is overfitting as occurred.

Now, after we have grown the full tree we had using that reduced error pruning algorithm to post-prune and as we are post-pruning the number of nodes in the tree is decreasing and we see that accuracy is increasing up to this point, when we have about 18 nodes. The accuracy has increased after that further pruning does not give you better accuracy. So, we stop here. This is the effect of post-pruning on the accuracy of decision.

(Refer Slide Time: 21:18)



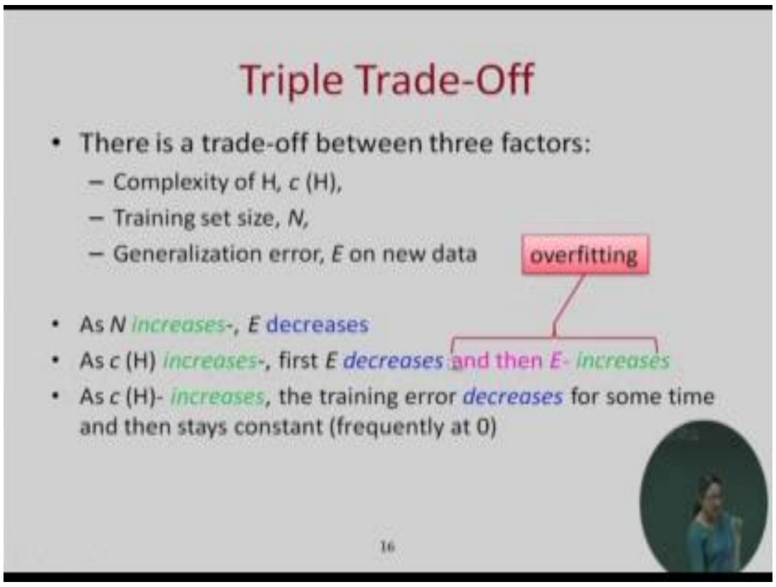
Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than *C* or *f*
- Underfitting: H less complex than *C* or *f*

15

Now, we have discussed earlier that learning involves generalization or induction and it is an ill-posed problem because you do not have sufficient data. So, we need some bias to restrict the hypothesis space or put some preferences, so that you want to prevent overfitting.

(Refer Slide Time: 21:45)



Triple Trade-Off

- There is a trade-off between three factors:
  - Complexity of H, *c* (H),
  - Training set size, *N*,
  - Generalization error, *E* on new data
- As *N* *increases*-, *E* decreases
- As *c* (H) *increases*-, first *E* *decreases* and then *E*- *increases*
- As *c* (H)- *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

overfitting

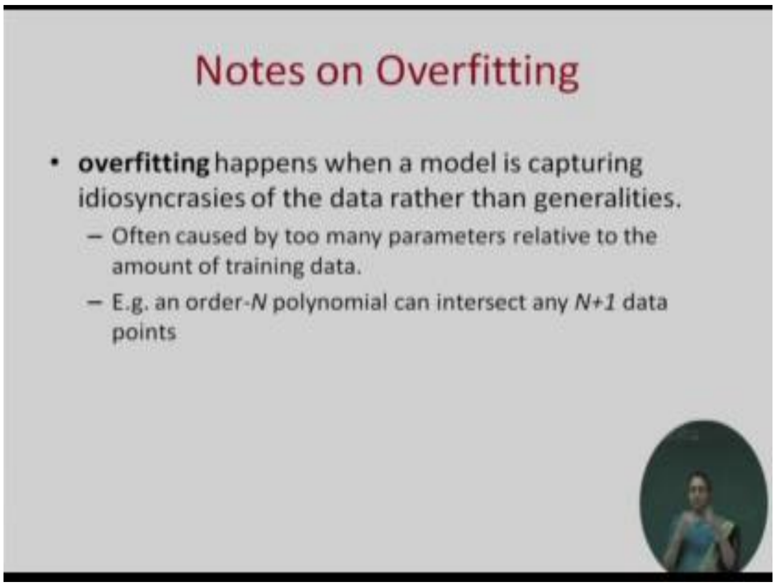
16

And there are three trade-offs which are involved. These trade-off include these three factors the complexity of the hypothesis, a more complex hypothesis done is more flexible it can accommodate more powerful functions, but more complex hypothesis are may lead to overfitting.

The second get training size if training size is too small then using that training set you cannot come up with the good function they may be overfitting. The third is generalization error that is the true error the error on a on the population or unseen instance. So, we want to look at the trade-off involving these three factors the complexity of the hypothesis, the size of the training set and the generalization error. We find out that as N increase as training size increases the error decreases.

As complexity of hypothesis increases first, the error decreases then the error increase as we saw in the decision tree as we are growing the decision tree. Initially error is decreasing then it is increasing and as complexity of hypothesis increases the training error decreases for some time and after sometime it may become 0 it become statistic, but the test error or the true error first decreases and then increases and this is when error starts increasing, this is the region where we say overfitting is occurred, this is where overfitting as occurred.

(Refer Slide Time: 23:37)



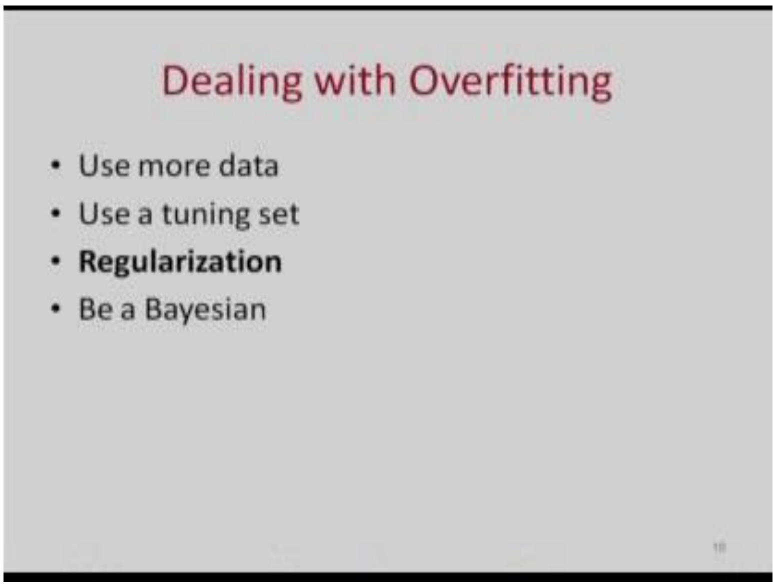
Notes on Overfitting

- **overfitting** happens when a model is capturing idiosyncrasies of the data rather than generalities.
  - Often caused by too many parameters relative to the amount of training data.
  - E.g. an order-*N* polynomial can intersect any *N+1* data points

So, overfitting is happening when a model captures the idiosyncrasies of the data rather than the generalities in the data. If you have too many parameters, if you have too many nodes in the decision tree, if you have too many parameters overfitting can happen, for examples, we discussed about regression in an earlier class. If you have linear regression you have few parameter beta 0 and beta 1, beta 2, but if you have second degree polynomial, fifth degree polynomial, tenth degree polynomial, you have many more parameters and overfitting is more likely to happen because an Nth order polynomial can intersect any N plus 1 data points.

If you have two few data points you should use a function which has fewer parameters because otherwise overfitting can occur. We have seen how overfitting can be dealt within decision tree in regression also overfitting can occur.

(Refer Slide Time: 24:27)



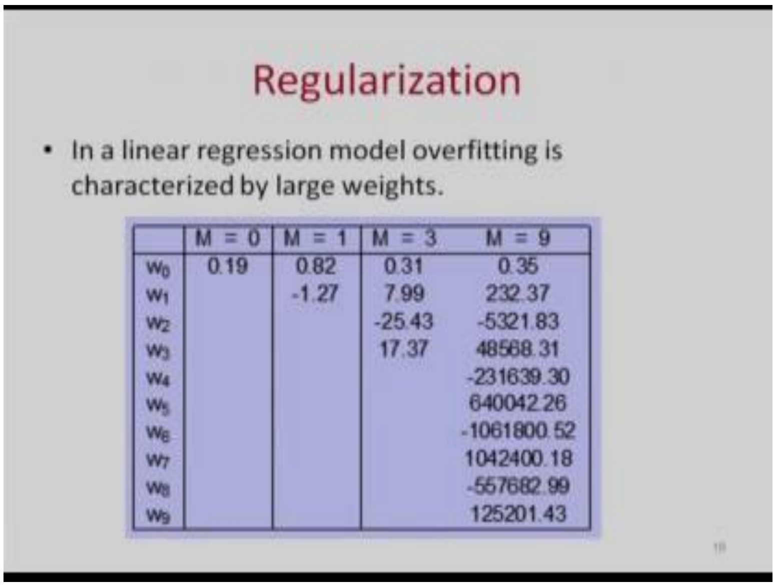
Dealing with Overfitting

- Use more data
- Use a tuning set
- **Regularization**
- Be a Bayesian

And in regression or linear regression, overfitting can occur and if you want to deal with overfitting there as some standard ways of dealing with overfitting. First of all in any of any learning algorithm if you can get more data that is one way of dealing with overfitting.

Now, second is the idea of cross validation using the validation set to decide you know what is happening? So, this is a very general approach to deal with overfitting and then we will later look at Bayesian approach overfitting, but one approach will just want to spend of two minutes on is Regularization. Regularization is a method of dealing with overfitting in algorithms like linear regression.

(Refer Slide Time: 25:15)



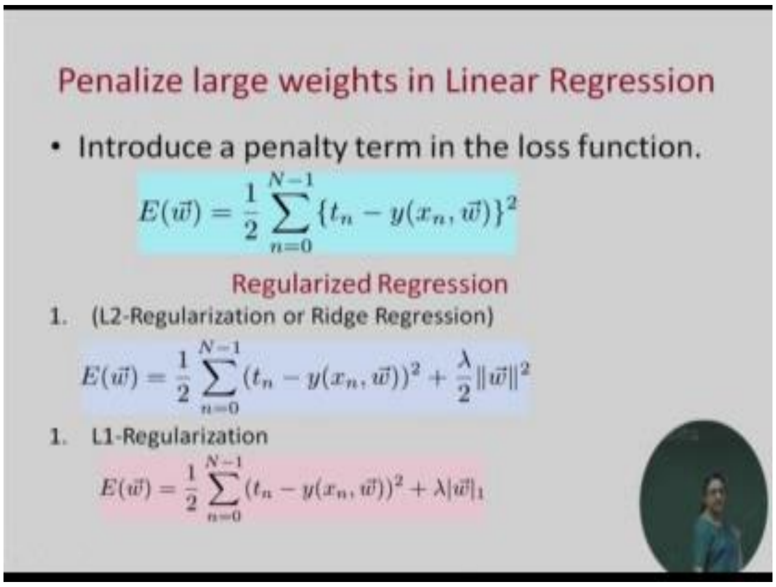
Regularization

- In a linear regression model overfitting is characterized by large weights.

	M = 0	M = 1	M = 3	M = 9
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Now, in a linear regression model, we typically overfitting are characterized by large weights of the coefficients. The coefficient beta 0, beta 1, beta 2 etcetera that we use on linear regression, if overfitting is occurs normally this values becomes too large. So, to deal with overfitting, if you look at this slide here these shows the values of the different coefficients at different instances and as you see for this is for 0th degree polynomial, this is 1 degree polynomial, 3 degree polynomial, 9th degree polynomial. When we fit the data with 9th degree polynomial, we have seen that the weight is very coefficients are very large and this is an indicator of overfitting.

(Refer Slide Time: 26:10)



Penalize large weights in Linear Regression

- Introduce a penalty term in the loss function.

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}\{t_n - y(x_n, \vec{w})\}^2$$

Regularized Regression

1. (L2-Regularization or Ridge Regression)

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \frac{\lambda}{2}\|\vec{w}\|^2$$

1. L1-Regularization

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \lambda|\vec{w}|_1$$

So, in order to prevent overfitting, one of the methods that we can do with polynomial regression or even a linear regression is use regularization and one way of regularization is to penalize large weights. So, earlier we have seen that the error function which we try to optimise is the sum of squared errors that is value of y minus y at whole square sum over all the examples. So, tn is the target value and this is the value of the predicted value. Now, if you want to deal with overfitting in addition to this in the error function you can add another term based on the size of the coefficient right. So, there a different types of regularization L1 regularization, L2 regularization etcetera.

For linear regression, there are two very popular types of regularization method, L2 regularization or which regression it adds a factor lambda by 2 w square, where w square is the L2 node of the coefficients and in L1 regularization or lasso, we use the L1 node. So, this is added to the penalty function and this is function that you try to optimise. So, you prefer the weights, so that the weights are small and the error is small if we do that it can help in preventing overfitting in linear regression. With this we come to an end to this class.

Thank you.